A Wholly Owned Subsidiary of
American Systems Corporation

7 Characteristics of Dysfunctional Software Projects

Mike Evans, Alex Abela, Tom Beltz

www.iceincusa.com
(757) 463-8483 x 24

Agenda

- Who We Are
- What Our Article Was Based on
- Why This Information Is Important
- What We Did
- Lessons Learned
- How This Applies to You and What You Can Do About It

Who We Are

- Integrated Computer Engineering (ICE), Inc. is a wholly owned subsidiary of American Systems Corporation
- Among other things, we help our clients identify and evaluate project risk and show them how to mange them
- Over the last 12 years we have assessed more than 300 Federal, DoD, State and commercial system acquisition and software development projects

The Best Reason of All

"Risk factors are always present that can negatively impact the development process, and if neglected, can tumble you unwittingly into program failure. To counteract these forces, you **must actively assess, control, and reduce software risk on a routine basis**."

Staff Writer, USAF STSC

Assessments & Tools Address More Than Just Software Projects

- We apply a number of assessment models and tools based on the client's requirements and the scope and objective of the assessment:
  - Government Program Office (GPO) Assessment
  - CMM Equivalency Assessment
  - SA-CMM Equivalency Assessment
  - Developers Capability and Risk Assessment
  - Institutional Risk Assessment
  - Enterprise Risk Identification
  - Project Risk Identification Process
  - Risk Treeing
  - ICE Historical Risk Data Base
- A primary goal of all assessments is to identify, categorize, document and report project risks
  - A "seeded" Risk Radar™ database is delivered, if requested

What Was Our Article Was Based On

- Several years ago, we observed patterns emerging among troubled software-intensive projects
- We began collecting, collating, and analyzing risks and indicators from these “dysfunctional” projects and documented common threads between them
- Financed through corporate R&D resources, our effort was constrained by limited funding

Additional Study Resource Data

- Our initial study incorporated data from other sources, including:
  - The Institute for Defense Analysis
  - Risks identified by Capers Jones
  - Related work performed by ICE, Inc. in support of the Software Program Managers Network (SPMN)
  - Risks and project characteristics identified by Tom DeMarco in his great book, ”Why Does Software Cost So Much”

Our Risk Database

- Our database grew to more than 890 primary and secondary risks, project indicators and risk mitigation activities
- This data was analyzed to identify any common causes and repetitive behaviors that seemed to precipitate the transition of risks into problems
- We focused on the primary risks since the secondary risks and indicators branched out from these

Why Is This Information Is Important

- Provides an alternate perspective of program performance
  - Not based on what you do or how you do it, but rather from the vantage point of program culture and attitude
- The customary measures of a project's potential for success don't fully apply here
  - If you ignore reality and simply hope or assume that your project will succeed, you'll be unable address critical issues in a timely manner
  - If effective management processes are not applied, the chances are that any process improvement initiatives you undertake will not succeed
  - Without effective and proactive program management leadership, there is no common purpose, goals, or measures of success

What We Did

- Began as a “doodle” on a flight from Washington to San Jose
- Initial common risk indicators identified:
  - “We’ll probably be O.K. if we can get those 5 new people on board.”
  - “The hardware will be delivered on time…Won’t it?”
  - “We can get it coded without the full design.”
- When these observations were introduced into the project environment, they became, “We’re O.K. because Charlie has it under control.”
- Reality was ignored, processes were modified and unrealistic assumptions became project decision drivers. Denial had taken hold.
- Performed “Mini Fault Tree Analysis” to look for other common issues:
  - What can go wrong followed by, what are the causes?
  - Began with 17 assumptions and ended with 7

Findings

- 7 Dysfunctional Areas:
  - Unwarranted optimism and unrealistic executive management expectations
  - Late decision-making
  - Inappropriate use of the standard software process
  - Missing or inadequately implemented program activities
  - Lack of leadership
  - Early declarations of victory
  - Absence of Risk Management

Analytical Results

Characteristic	Number of risk events applicable to specific Risk Designator	Number of risk events for Characteristic	Frequency of occurrence relative to all observed risk events (See Note 1)
1. Failure to Apply Essential Project Management Practices	246	480	57%
	6		
	36		
	66		
	10		
	116		
2. Unwarranted Optimism and Unrealistic Executive Management expectations	246	344	41%
	32		
	66		
3. Failure to Implement Effective Software Processes	162	248	30%
	15		
	26		
	45		
4. Premature Declarations of Victory.	116	165	20%
	46		
	3		
5. Lack of Program Management Leadership.	66	106	13%
	3		
	5		
	32		
6. Untimely Decision-making	15	63	8%
	3		
	45		
7. Lack of Proactive Risk Management	4	24	3%
	9		
	11		

What the Data Told Us

- One facet of the problem is addressed through various, and very effective, process improvement initiatives:
  - CMM, CMMI, SA-CMM
  - SPMN 16 Point Plan
  - Company Initiatives

- Additional focus must be on attitudes and culture:
  - Managers can't deny reality
  - You can't mandate success
  - Risks will transition to problems if you ignore them
  - Hoping to meet a milestone won't get you there
  - Attitude is as much of a "success breaker" as process effectiveness

Unwarranted Optimism & Unrealistic Executive Management Expectations

- Managers and project staff recognize the potential for impending impacts, but assume that things will work out:
  - "Not knowing what you don't know" causes project staff and management to think they can accomplish what they cannot
  - Manager's flock to unproven or unrealistic "silver bullets", rather than addressing the true nature of the project
  - Managers who are only tuned into promises of success and fail to accept realistic predictions that do not meet their expectations will not be provided with true reports on the status of programs, until a crises is actually in progress
  - They waste the resource they can't replace, "Time."
- The staff is not capable of implementing the product and applying the technologies selected
  - Excessive turnover may impact project success
- Project plans are unrealistic, or not implemented and do not result in a predictable development environment

Optimism Is Normal & Essential; Needs to Reflect Reality

- There is an underlying belief that all will be well
  - "We are confident that the Software Gods will shine upon this deserving project."
  - When managing or participating in a software intensive project there is no reason to be optimistic; history doesn't support it

- Staff with insufficient experience may unaware of the magnitude of the tasks or the problems they are attempting to solve;
  - They have oversimplified the ease of achieving the required outcomes;
  - They attempt to implement "silver bullet" or unproven technology solutions without having evaluated their effectiveness or impact

Late Decision-making

- Managers often delay making key decisions or taking required action even in the face of overwhelming evidence of impending catastrophe.
  - Managers often fail to grasp the urgency required to address these issues hoping that things will right themselves before the issue or problem becomes visible.
- Management is the art of planning work, so that it can be accomplished within constraints of time, cost, and other resources at a level that will be competitive in the marketplace.
  - Delays caused by slow decision makers erode these constraints while staff waits for a direction or resources.
  - Risks, which become problems often are the result of preventive actions, engineering steps, or management practices not implemented in time.
    - Plans not in place in time
    - Need for them not recognized due to late completion of plans, standards or engineering process.

Slow Decision Making Stops Progress

- "You can fix a bad decision, but no action occurs while projects wait for managers to decide what to do."
- What do those waiting for the decision do with their time?
- Time spent waiting for management decisions erodes the available project time and resources, raises risk and demoralizes the project staff.

Inappropriate Use of the Standard Software Process

- Programs often migrate to “state of the art” or “flash” technologies without an understanding of:
  - cost of bringing them into their environment;
  - problems associated with adapting tools or methods to their culture;
  - the risk that the tool or technology may be inconsistent with the development requirements or constraints.

- Many managers of software projects assume that since the project is staffed by trained software engineer’s project specific standards, guidelines and common tools are unnecessary.
  - “After all, we’re a Level 3 organization and of course we know how to develop software.”

Factors Impacting Application of Common Processes

- Project uniqueness
  - To paraphrase Tim Lister; each project is unique. Each has it's own quirky clients. It's own unique staff, it's own expectations of success. Could it be that adaptation of process is 90% of the problem and the common processes are marginal?
  - Technology is not a "cookie - cutter" solution to every development problem. Neither are the common processes so painfully defined in order to achieve an acceptable CMM Level.

- Project balance
  - Technology, tools, process and people to implement the project plan all must be in balance and this balance has to be reached at the project, not the organizational level.

Factors Impacting Application of Common Processes (Cont.)

- While common processes that are critical to repeatable success, as defined by the CMM, the unique adaptations of these to project environments are the ultimate measure.
  - Repeatability, once the process is defined, is not a factor in assuring specific program success.

Missing or Inadequately Implemented Program Activities

- Essential “non tech” project disciplines (estimation, scheduling, logistics, manpower loading, the management of people, configuration management, and others) are not viewed as part of the essential engineering core of the project.
  - Many of these project areas tend to provide the engineering-centric manager with more reality then they can handle.
  - Many managers look on these tasks as non-essential bureaucratic activities even though the non-engineering disciplines provide the essential core of the project and are critical to the reduction of project risk.

"Low Tech" Tasks Aren't Always Addressed

- While the mainstream software tasks have been reasonably well planed and implemented, specific methods have not been implemented to deal with unique project commitments, risks and operational requirements

- These methods do not fall in the experience mix of those on the project.
  - How often have we observed projects that do not engineer reliability into a software product even though they are facing a 24/7/365 operational requirement because they haven't done this type of engineering in the past?
  - They try to test it in at the end, often to no avail.

Must Do's

- Management and engineers often see several methods as cultural pariahs

- Out of the class of methods used in software project, a key subset is not at the top of any manager's list of "must do's".
  - Risk management
  - Configuration management
  - Earned value reporting
  - Metrics
  - Re-estimation
  - Quality assurance
  - Inspections
  - Rigorous testing

Common Characteristics

- What pushes these “Must Do’s” off of the top-ten list?
  - They are perceived as bureaucratic having the potential of getting in the way of real engineering.
  - They are perceived as “low tech”; not in the project mainstream”
  - Methods such as risk management, metrics, re-estimation often provide management more reality than they can deal with in the normal course of a project.
  - These methods remove the potential for denial.

- By having an early warning, there is an opportunity to mitigate potential effects or even avoid the effects altogether.

Lack of Leadership

- Managing a software project requires courageous, and often clairvoyant leaders who are willing the confront today's challenges to avoid tomorrows catastrophes.
- Attributes of a good manager include:
  - having a broad range of technical software development experience and domain knowledge.
  - The ability to:
    - manage people;
    - manage the dynamics of a team environment;
    - recognize project and staff dysfunction;
    - lead so the expected or essential result is achieved
- Many software project managers were promoted from within and have no training other than their engineering background

Two Types of Problem Managers

- One has the requisite technical and domain level experience and in some instances excel in these attributes, but fails to have the leadership, personnel management and programmatic skills required.
  - "problem managers" generally fail to focus on meeting the high level objectives of the program, dwelling instead in their area of familiarity and expertise, such as writing good code.
- The second is an individual with limited or no understanding of the essentials of software engineering and who is promoted to this role from some other discipline (e.g., Finance, QA, HR).
  - This type of manager might trade off essential engineering steps to save cost, improve image or convince others that the project can accomplish something it cannot
- Potential result of these "problem managers" is, at best, a project that drifts to some conclusion or at worst, a project that drifts into chaos

Management Experience Must Match Project Scale

- "Poor project management will defeat good engineering, and is the most frequent cause of project failure." (Watts Humphrey)
- Too many people who are making decisions as to how software is to be developed have never developed software.
- Compounding this is the fact that many managers of large scale software projects were trained in an academic environment that focuses on the coding of small scale, 1-semester projects.
- While their management skills may be adequate and they have extensive experience participating or managing non-software intensive projects
  - The lack of specific experience managing large scale software projects with many creative engineers developing many intangible products probably will become a problem
  - Probably impact the success of the project.
  - They think in the small while the reality of the project is in the large.

Premature Declarations of Victory

- Many projects rely on qualitative or subjective measures of success
  - "The 90% completion factor"
  - "I'll know it when I see it"
  - "All indications are we'll be done by Friday"
  - "I'm optimistic that we're about to turn the corner"

- All paint an unrealistic view of the true state of the project and the risks of reaching an agreed to conclusion.

Success Factors

- Success cannot be declared until the agreed to products have been completed and delivered:
  - the negotiated quality targets are met
  - the documentation is complete
  - the customers are satisfied through observations, analysis and demonstrations that the product will satisfy their needs.

- Effective Quality Assurance, testing, metrics and on-going quality control provides these assurances.

We Often Do It To Ourselves

- "If you didn't know that few managers receive any management training at all,
  - you might think there was a school they all went to for an intensive course on Parkinson's Law [Work expands to fill the time allocated] and it's ramifications …
  - It gives them the strongest possible conviction that the only way to get work done at all is to set an impossibly optimistic delivery date." - DeMarco
- Managers set themselves up by setting unrealistic commitments and then can only achieve success by shaving the process or negotiating away important product characteristics.
  - Managers jeopardize product quality by setting unreachable deadlines.
- The reality is that a project that trades off it's basic commitments to quality to deliver a product, set themselves up for far more difficult problems later on;
  - a disgusted customer.

Lack of Risk Management

- "What distinguishes the best organizations and best managers is not just how well they do in their successful efforts, but how well they contain their failures." – DeMarco
  - The best managers of software projects seem to have an uncanny ability to anticipate what can happen to their project and come up with a "just in time" mitigation to avoid the full impact of the problem.
    - There is no magic to this skill; It is just effective risk management being applied to the problem of managing software.
  - Seeing a potential risk and ignoring the possible impact is, an unfortunate trait of far too many software projects we have assessed.

- By all accounts effective risk management is the one common factor in successful projects

Project Balance

- "The problem of project management, like that of most management [is] to find an acceptable balance among time, cost and performance."
  - When a project moves out of balance, a risk results.
  - Often schedule performance becomes most important due to customer pressures so cost and product performance lose emphasis.
  - Often the product takes center stage due to a customer review so cost and schedule performance focus drifts into the shadows.
- "An effective risk - management program is dynamic and ongoing throughout the development process and requires the participation of everyone involved."

Assess Risk Management Effectiveness – Not Just Process

- We spend a significant time during our assessments exploring the degree and effectiveness that a project has utilized risk management as part of it's management structure.

- This assessment area, in our minds, is a dead on indicator of the potential overall success of the project.
  - Projects that do an ineffective job of managing risk seem to be constantly reacting to problems that are triggering off while those that manage risk well anticipate rather than react.
  - To maximize potential for success, risk management should play a visible and key role in the process of project management.

Managers Often Cannot Tolerate Risk

- After assessing over 300 projects and actively working with many on improving risk management process and making them more effective in their unique cultures we believe that a basic reality gets in the way of effective application;

- Managers don't really want to know what risk management tells them.
  - There are a significant number of things that can get in the way of success.
  - Can do attitudes may not really put us over the bar.
    - All Projects needs at least on "can't do guy"; The Risk Manager

The Risk Process Syndrome

- While every project we assess professes to implement risk management we have observed two very different application focuses, the process focus and the cultural imperative.
  - The process focus has an identified risk manager who busily sees that the seven steps of risk management, Identify, Analyze, Prioritize, Plan, Implement, and Evaluate
  - Documents are visible in the project to all stakeholders.
  - Managers never use the actual risks to influence decisions or plot a future course for the project.

The Risk Process Syndrome (Cont.)

- The Risk Manager has built a close looped system where actual risks never leak out. We have observed situations where even the types of risks that can be identified are dictated.
- During one assessment an engineer passed on instructions he had received from the Risk Manager of a major commercial program,
  - "Don't give me any cost schedule or process risks because, if they got out they will make the project look bad."
  - That Risk Manager was more concerned with process integrity than the result of the process.

- Very few projects we assess, probably they could be counted on one hand, implement risk management as a cultural imperative.

How Does This Apply To You And What Can You Do About It

- Why don't projects address these issues if they are so apparent?
  - Two reasons, denial and cultural.
    - Denial in the sense when you are fighting the day to day realities of a software intensive project it is very easy to assume that "The indicators of disaster are probably wrong and we won't be impacted the way the other 12 projects were."
      - Denial is the excuse that enables software managers, or any managers for that matter, to do stupid things.

How Does This Apply To You (Cont.)

- Cultural problems are harder to solve than technical problem.
  - Addressing these problems means that a manager has to understand what makes his or her project tick;
    - How do staff interact,
    - what motivates them,
    - why don't they address important issues even though they are essential to project success.
- Only then can the project effects of these seven factors be understood and minimized. For an untrained manager, or a manager under pressure, this is a hard pill.

- Yesterdays problems are today's risk-Believe it!

Assessment Trends

- The 7 characteristics we identified seem to be evident, all or in part, in most of the projects we assess.
- By analyzing the information in the Observed Risk Database from different biases or perceptions, we can find different “truths” and these would be no less valid.
- Our observations are backed up by the experiences of all ICE, Inc. assessors

Additional Sources

- Additional Sources Feeding Risk Data Base:
  Institute for Defense Analysis (IDA). Technical Risk Indicators For Embedded Software Development. IDA Paper P-3027, October, 1994.
- Jones, Capers T. Assessment and Control of Software Risks. New Jersey: Yourdon Press, Englewood Cliffs, February, 1994.
- Integrated Computer Engineering, Inc. ICE Disk. Version 1.0 Arlington VA: ICE, 2000.
- DeMarco Tom, Why Does Software Cost So Much, Dorset House Publishing, New York, 1995
- Software Program Managers Network (SPMN). 16 Critical Software Practices For Implementing Performance-Based Management. Version 3.0, Arlington VA: Integrated Computer Engineering, Inc., 2 August, 2000.

References

- Technical Risk Indicators for Embedded Software Development. Institute for Defense Analysis, Paper P 3027,
- Jones,Capers T. Assessment and Control of Software Risks. New Jersey: Yourdon Press,Feb.1994.
- DeMarco Tom. Why Does Software Cost So Much? New York: Dorset House Publishing,1995.
- Software Program Managers Network.16 Critical Software Practices For Implementing Performance-
- Based Management.Ver.3.0, Arlington, Va.: Integrated Computer Engineering,Inc.,2 Aug.2000.
- Standish Group International. “Chaos.” Open Computing. Copyright Mar.1995 SPC.
- Jones,Capers T. Assessment and Control of Software Risks. New Jersey: Prentice Hall,Feb.1994.158.
- Lister,Tim.“Software Management for Adults.” Software Technology Conference,1996.
- Davis,Alan.“Software Lemmingineering.” IEEE Software Sept.1993.
- .Humphrey,Watts.“Three Dimensions of Process Improvement: Part I: Process Improvement.” CrossTalk Feb.1998.
- Putnam,Lawrence H.,and Ware Meyers. Industrial Strength Software, Effective Management Using Measurement. Los Alamitos, Calif.: IEEE Computer Society Press,1996.1.
- P.V.Norden.Useful Tools For Project Management, Operations Research in Research and Development. Edited by B.V.Dean. New York: John Wiley & Sons,1963.
- Molt,George.“Risk Management Fundamentals in Software Development.” CrossTalk Aug.2000.
- Boehm,Barry,Raymond Madachy, and Chris Abts. “Future Trends: Implications in Cost Estimation
- Models.” CrossTalk Apr.2000.
- Hall,Elaine.Managing Risk. Reading Mass.: Addison Wesley 1997.5.
- .DeMarco,Tom,and Tim Lister. Peopleware, Productive Projects and Teams 2nd ed. New York: Dorset House Publishing,1999.4.

Acronyms

ASC	American Systems Corporation
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integrated
GPO	Government Program Office
HR	Human Resources
ICE	Integrated Computer Engineering, Inc.
IDA	Institute for Defense Analysis
QA	Quality Assurance
R&D	Research & Development
SA-CMM	Software Acquisition Capability Maturity Model
SPMN	Software Program Managers Network